READ THIS NOW!

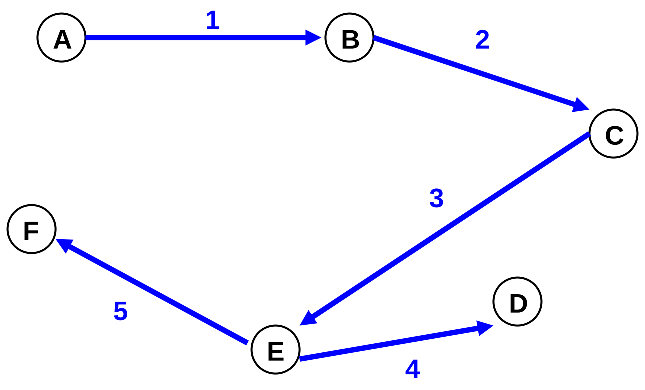
- Print your name in the space provided below.
- Unless a question involves determining whether given Java code is syntactically correct, assume that it is, and that any necessary `import`s have been made.
- There are 10 short-answer questions, priced as marked. The maximum score is 100.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your fact sheet.
- Aside from the allowed one-page fact sheet, this is a closed-book, closed-notes examination.
- No laptops, calculators, cell phones or other electronic devices may be used during this examination.
- You may not discuss this examination with any student who has not taken it.
- Failure to adhere to any of these restrictions is an Honor Code violation.

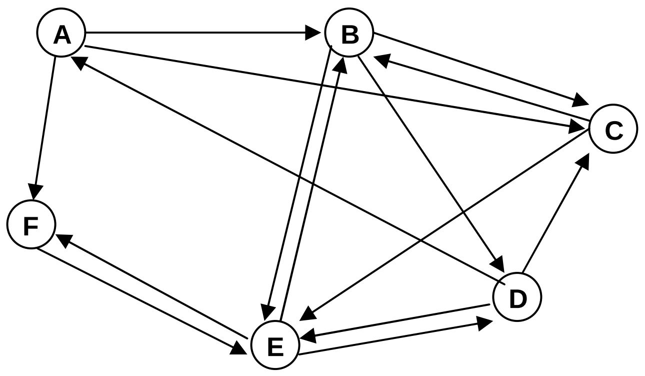
Name (Last, First) ____
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

____
signed

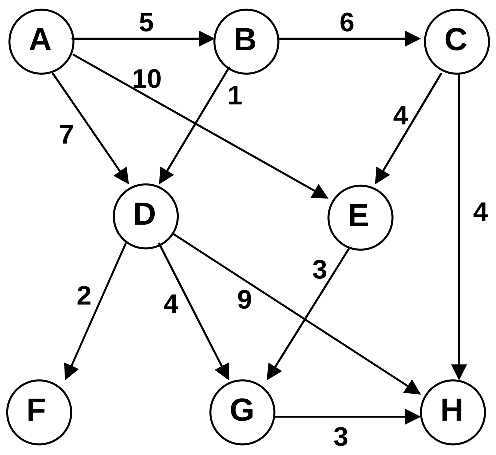
1. [12 points] Draw a new graph showing the spanning tree for the graph below obtained by applying a depth-first traversal starting at node **A**. Visit neighbors in alphabetical order, and number the edges in the order they are added to the tree.





2. [12 points] Apply Dijkstra's SSAD algorithm to the graph below. Show work and list the distance found for each vertex in the graph. Start at vertex **A**.

A	B	C	D	E	F	G	H
add	5	x	7	10	x	x	x
	add	11	6	10	x	x	x
		11	add	10	8	10	15
		11		10	add	10	15
		11		add		10	15
		11				add	13
		add					13
							add
0	5	11	6	10	8	10	13



3. [10 points] Given a node **X** in a directed graph, define a distance-k successors of **X** to be a node **Y** such that the shortest path from **X** to **Y** in the graph is of length *k* (i.e, the path contains *k* edges). So, in the graph given in question 1, the distance-2 successors of the node **A** would be **D** and **E**.

Assuming the directed graph has 100 nodes and 1000 edges, which representation (adjacency matrix or adjacency list) would provide the most efficient support for the operation of finding the distance-*k* successors of randomly-selected nodes in the graph? The goal here is to minimize the average time cost, and the space cost is not an issue at all.

Justify your conclusion carefully.

Since there are 1000 edges and 100 nodes, the average number of edges going out from a node must be 1000/100, or 10.

For the adjacency matrix:

distance-1	**traverse a row of the matrix**	**100**
distance-2	**traverse a row, finding 10 neighbors and then traverse the rows for those 10 neighbors; so cost of distance-1 + traversing 10 rows**	**100 + 10*100**
distance-3	**cost of distance-2 + traversing 10^2 rows**	**100 + 10*100 + 10*10*100**

So, it appears that the cost of finding the distance-k successors of a node would be about:

100 * (1 + 10 + 10^2 + . . . + 10^k)

Aside: if you raise the matrix to powers, the resulting matrices indicate the level-1, level-2, etc., neighbors for each vertex; however, the matrix multiplication is Θ(N^3).

For the adjacency list:

distance-1	**traverse a neighbor list**	**10**
distance-2	**traverse a neighbor list, finding 10 neighbors and then traverse the neighbor lists for those 10 neighbors; so cost of distance-1 + traversing 10 neighbor lists**	**10 + 10*10**
distance-3	**cost of distance-2 + traversing 10^2 neighbor lists**	**10 + 10*10 + 10*10*10**

So, it appears that the cost of finding the distance-k successors of a node would be about:

10 * (1 + 10 + 10^2 + . . . + 10^k)

Conclusion: the cost would be MUCH smaller for the adjacency list structure, a factor of 10 smaller in fact!

4. [14 points] Consider a hash table consisting of $N = 10$ slots, and suppose integer key values are hashed into the table using the hash function:

$$H(Key) = \text{the sum of the digits of } Key$$

Suppose that collisions are resolved using quadratic probing.

Show the contents of the hash table after the following key values have been inserted in the given order:

297 749 876 897 266 752 451

If you want partial credit, show any calculations you perform.

Slot number	Contents
0	**749**
1	**876**
2	
3	**752**
4	**897**
5	**266**
6	
7	
8	**297**
9	**451**

Example:
H(297) == 18 so the home slot is 18 % 10 == 8

H(749) == 20 so the home slot is 20 % 10 == 0

H(876) == 21 so the home slot is 21 % 10 == 1

H(897) == 24 so the home slot is 24 % 10 == 4

H(266) == 14 so the home slot is 14 % 10 == 4 full!
probe to 4 + 1 ^2== 5

H(752) == 14 so the home slot is 14 % 10 == 4 full!
probe to 4 + 1^2 == 5 full!
probe to 4 + 2^2 == 8 full!
probe to 4 + 3^3 == 3

H(451) == 10 so the home slot is 10 % 10 == 0 full!
probe to 0 + 1^2 == 1 full!
probe to 0 + 2^4 == 8 full!
probe to 0 + 3^2 == 9

5. [8 points] When a hash table is searched for an element **X**, under what logical condition can the search be terminated with the conclusion that **X** is not in the table? Justify your conclusion.

If the search finds a slot that is EMPTY, we can correctly conclude that X is not in the table, because the insertion of X would not have probed past that EMPTY slot.

If the search finds a TOMBSTONE or FULL slot (not holding X), we cannot conclude that X is not in the table, because we know that there was once an element in the tombstoned slot and therefore the insertion of X might have probed past that slot.

6. [10 points] Consider designing a hash function to be used in organizing a collection of phone company customer records. Each customer has a unique ID, of the form `<citycode>-<numeric>`, where the city code consists of three to five upper-case letters and the numeric part is a five-digit number in the range 00000 to 99999. For example: ROA-37412 or BBURG-23512. The company has hundreds or more customers in each particular city, but no more than 99999, and the numeric part is assigned sequentially as new customers are added.

The hash function will be applied to customer IDs. The following hash scheme is suggested: represent each character in the city code by its ascii value and add those values together, then take the left-most three digits of the numeric part of the ID and add that in, and finally mod the result by the size of the hash table.

The computation may result in an integer overflow, but that is not objectionable. The number of actual customer IDs is known, and there is no concern about the amount of space needed for the table.

Give two good, unrelated reasons why this hashing scheme should not be adopted unless it is modified.

The city code is hashed in a way that does not take the order of the characters into account.

A good hash function would be sensitive to the ordering of the characters.

If there are only hundreds of customers in a city, then the left-most three digits of the numeric code may not show much variation at all. In fact, the first one or two could be the same for every customer in that city.

A good hash function would be sure to use the parts of the key that vary the most.

Aside: if you cited arithmetic overflow, you ignored both the problem statement and the fact that if the city code is limited to 5 characters (as stated) there is no possibility of overflow anyway. In fact, the relatively small range of possible values could also be cited as an objection.

7. [10 points] What is the single most important property of B-trees (including the variations like the B+ tree) that make them more suitable for on-disk storage than the other kinds of trees we have discussed? Explain why.

Each B-tree node stores lots of key values, so fetching a single B-tree node from disk gives us lots of data to work with to determine the direction of the search, and fetching a larger B-tree node from disk will take only a little more time than fetching a tiny AVL or BST node from disk.

8. [8 points] Consider building a skiplist of 16 nodes holding the integer values 0 through 15. What would the average search performance be like if every node in the skiplist was in level 2 (i.e., had 3 forward pointers)? Justify your conclusion.

Since every node is in the same level, this is effectively just a singly-linked list and the expected search cost will be $\Theta(N)$.

9. [8 points] What is it about the rotation operations used in an AVL tree that guarantees the height of the tree is limited to $\Theta(\log N)$? Note: this is a question about the rotations, not about the tree.

There are two factors that are important:

- **a rotation will, in many cases, result in a tree that is shorter than a BST created from the same sequence of operations**
- **the rotations are cheap, $\Theta(1)$ in the worst case, and so the cost of the rebalancing doesn't wipe out the benefit of keeping the tree short**

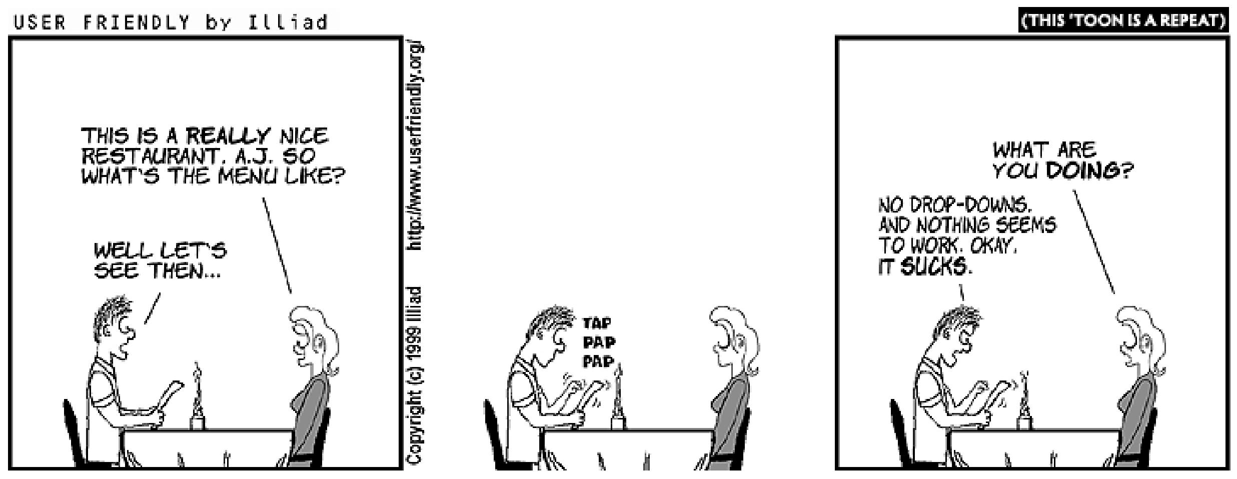
10. [8 points] Suppose a hard drive manufacturer wants to reduce the average and worst-case seek times for a hard drive design without reducing its capacity. Currently the drive uses *P* platters, with *T* tracks per surface and *S* sectors per track.

What aspect(s) of the hardware could be changed to achieve the goal, and how?

The average seek time depends on the head start time, the track-to-track seek time, and the number of tracks on a surface (since on average you'll cross 1/3 of them).

So, the average seek time could be reduced by:

- **decreasing the head start time (but this would be a very small gain)**
- **decreasing the track-to-track seek time**
- **decreasing the number of tracks per surface, and compensating by increasing the number of sectors per track or the number of platters (in order to avoid reducing the capacity)**


USER FRIENDLY by Illiad
(THIS 'TOON IS A REPEAT)
http://www.userfriendly.org/
Copyright (c) 1999 Illiad
THIS IS A REALLY NICE RESTAURANT, A.J. SO WHAT'S THE MENU LIKE?
WELL LET'S SEE THEN...
TAP PAP PAP
WHAT ARE YOU DOING?
NO DROP-DOWNS. AND NOTHING SEEMS TO WORK. OKAY, IT SUCKS.